We claim:

1. A computer system for facilitating distributed directory-enabled applications using an eXtensible Markup Language ("XML") application program interface, the system comprising:

at least one processor;

at least one memory accessible to the processor;

a first application stored in a first portion of the memory;

a second application stored in a second portion of the memory;

software for an event system, the software comprising instructions for publishing an event by either the first or second application, subscribing to the event by the other application, and acting on the event by the other application, whereby the first and second applications interact with each other through the event system;

software for parsing XML files for the first and second applications, the software comprising instructions for accepting an XML file as an input stream, parsing the input stream, dynamically loading system services referenced in the input stream, and configuring the services; and

software for bridging, the software for comprising instructions for thread safeness, whereby a bridge utilizes semaphore access control to control thread access, smart pointers, whereby the bridge automatically manages the memory it requires, and opaque interfaces, whereby the bridge maintains interface compatibility when implementation changes occur in an interface.

2. The computer system of claim 1 wherein the first application is processed by a first processor and the second application is processed by a second processor.

3. The computer system of claim 1 further comprising a document object model utilizing numeric tags, wherein the application program interface communicates requests to the document object model using numeric tags.

4. The computer system of claim 3 wherein the application program interface accepts requests from the first and second applications in both numeric and string form, whereby the first and second applications may use either numeric or string based interfaces.

5. The computer system of claim 4 further comprising a tag management system comprising a tag manager, wherein the tag manager accepts string tags from the application program interface and converts the string tags to numeric tags for use by the document object model.

6. The computer system of claim 1 further comprising an optimized document object model, wherein the optimized document object model includes instructions for compacting a data tree comprising nodes and leafs by causing the nodes located directly above leaf level to contain the leafs.

7. The computer system of claim 6 wherein the optimized document object model includes instructions for dynamically expanding the leaf from the node when the data contained in the leaf is required.

8. The computer system of claim 1 further comprising a preprocessing stage for preprocessing one or more XML header files to generate and configure the bridge.

9. The computer system of claim 1 wherein the bridge uses XML to translate communications between the first and second applications when the first and second applications are developed in different programming languages.

10. The computer system of claim 1 further comprising a memory manager to assist the processor in memory management, wherein the memory manager includes instructions for controlling a memory pool for a thread.

11. The computer system of claim 10 wherein the memory manager includes instructions for controlling a plurality of memory pools for a plurality of threads.

12. The computer system of claim 11 further comprising a mechanism for appending free memory to a free list, whereby available memory for the threads is monitored by the memory manager.

13. The computer system of claim 1 further comprising one or more style sheet selection algorithms, whereby the first or second application produces a style sheet according to a browser type, a service type, and a class identification.

14. The computer system of claim 1 further comprising a cross-protocol query method, wherein multiple communication protocols are queried simultaneously by the first and second applications.

15. The computer system of claim 1 further comprising an object factory, wherein objects dynamically instantiated by the parser for the first or second application are used to instantiate a plurality of objects.

16. The computer system of claim 15 wherein dynamically instantiated objects default to a highest known applicable class.

17. The computer system of claim 15 further comprising a mechanism to cache objects by assigning an identifier to an object and retrieving the object when said object is requested by the first or second application, whereby caching may occur through the object factory.

18. The computer system of claim 1 further comprising an XML store, whereby data stored in a third portion of the memory accessible to at least one of the first and second applications is mapped between a database and at least one XML document, whereby the XML aspect of the document is retained.

19. A method for facilitating distributed directory-enabled applications using an eXtensible Markup Language ("XML") application program interface, the method comprising:

providing an event system, the event system comprising publishing an event by a first application or a second application, subscribing to the event by the other application, and acting on the event by the other application, whereby the first and second applications interact with each other through the event system;

parsing XML files for the first and second applications, the parsing comprising accepting an XML file as an input stream, parsing the input stream, dynamically loading system services referenced in the input stream, and configuring the services; and

bridging using thread safeness, whereby a bridge utilizes semaphore access control to control thread access, smart pointers, whereby the bridge automatically manages the memory it requires, and opaque interfaces, whereby the bridge maintains interface compatibility when implementation changes occur in an interface.

20. A software program for facilitating distributed directory-enabled applications using an eXtensible Markup Language ("XML") application program interface, the software comprising instructions for:

providing an event system, the event system comprising publishing an event by either a first application or a second application, subscribing to the event by the other application, and acting on the event by the other application, whereby the first and second applications interact with each other through the event system;

parsing XML files for the first and second applications, the parsing comprising accepting an XML file as an input stream, parsing the input stream, dynamically loading system services referenced in the input stream, and configuring the services; and

bridging using thread safeness, whereby a bridge utilizes semaphore access control to control thread access, smart pointers, whereby the bridge automatically manages the memory it requires, and opaque interfaces, whereby the bridge maintains interface compatibility when implementation changes occur in an interface.